CLAIMS:

1. A kernel emulator for non-native program modules, the emulator comprising:

an interceptor configured to intercept kernel calls from non-native program modules;

a call-converter configured to convert non-native kernel calls intercepted by the interceptor into native kernel calls.

2. An emulator as recited in claim 1, wherein the call-converter comprises a translator configured to translate a non-native paradigm for passing parameters into a native paradigm for passing parameters.

3. An emulator as recited in claim 1, wherein the call-converter comprises a translator configured to translate non-native CPU instructions into native CPU instructions.

4. An emulator as recited in claim 1, wherein the call-converter comprises a translator configured to translate addresses from non-native length into native length.

5. An emulator as recited in claim 1, wherein the call-converter comprises an argument-converter configured to convert non-native argument format into native argument format.

6. An emulator as recited in claim 1, wherein the call-converter comprises a translator configured to translate words from non-native word size into native word size.

7. An emulator as recited in claim 1 further comprising a memory constrainer configured to limit addressable memory to a range addressable by non-native program modules.

8. An emulator as recited in claim 1 further comprising a shared-memory manager configured to manage memory space that is accessible to both native and non-native program modules.

9. An emulator as recited in claim 1 further comprising a shared-memory manager configured to synchronize a native shared data structure with a non-native shared data structure.

10. An emulator as recited in claim 1 further comprising a shared-memory manager configured to manage memory space that is accessible to both native and non-native program modules, wherein the shared-memory manager maps versions of process shared data structures (SDSs) and versions of thread shared data structures (SDSs) between native and non-native program modules.

11. An operating system on a computer-readable medium, comprising:

a native kernel configured to receive calls from native program modules;

a kernel emulator as recited in claim 1 configured to receive calls from non-native program modules.

12. An operating system on a computer-readable medium, comprising:

a native kernel configured to receive calls from native APIs;

a kernel emulator as recited in claim 1 configured to receive calls from non-native APIs.

13. A method of emulating a kernel for non-native program modules, the method comprising:

intercepting kernel calls from non-native program modules;

converting the intercepted non-native kernel calls into native kernel calls.

14. A method as recited in claim 13, wherein the converting step comprises translating a non-native paradigm for passing parameters into a native paradigm for passing parameters.

15. A method as recited in claim 13, wherein the converting step comprises translating non-native CPU instructions into native CPU instructions.

16. A method as recited in claim 13, wherein the converting step comprises translating addresses from non-native length into native length.

17. A method as recited in claim 13, wherein the converting step comprises translating words from non-native word size into native word size.

18. A method as recited in claim 13 further comprising limiting addressable memory to a range addressable by non-native program modules.

19. A method as recited in claim 13 further comprising synchronizing a native shared data structure with a non-native shared data structure.

20. A method as recited in claim 13 further comprising mapping versions of process shared data structures (SDSs) between native and non-native program modules.

21. A method as recited in claim 19, wherein a process SDS of a native program module includes a pointer to a process SDS of a non-native program module.

22. A method as recited in claim 19, wherein a process SDS of a non-native program module includes a pointer to a process SDS of a native program module.

23. A method as recited in claim 13 further comprising mapping versions of thread shared data structures (SDSs) data structure between native and non-native program modules.

24. A method as recited in claim 22, wherein a thread SDS of a native program module includes a pointer to a thread SDS of a non-native program module.

25. A method as recited in claim 22, wherein a thread SDS of a non-native program module includes a pointer to a thread SDS of a native program module.

26. A computer comprising one or more computer-readable media having computer-executable instructions that, when executed by the computer, perform the method as recited in claim 13.

27. A computer-readable medium having computer-executable instructions that, when executed by a computer, performs the method as recited in claim 13.

28. An operating system embodied on a computer-readable medium having computer-executable instructions that, when executed by a computer, performs the method as recited in claim 13.

29. A method comprising:

determining whether an initiating program module is a native or non-native;

if the initiating program is non-native:

limiting available memory to a range that is addressable by the non-native program module;

establishing non-native a version of a shared memory data structure that may be synchronized with a native version of the same shared memory data structure.

30. A method as recited in claim 29 further comprising:

intercepting kernel calls from the non-native program module;

converting the intercepted non-native kernel calls into native kernel calls.

31. A method as recited in claim 29 further comprising emulating a non-native kernel for which kernel calls from the non-native program module are intended.

32. A computer comprising one or more computer-readable media having computer-executable instructions that, when executed by the computer, perform the method as recited in claim 29.

33. A computer-readable medium having computer-executable instructions that, when executed by a computer, performs the method as recited in claim 29.

34. A method comprising emulating a non-native kernel for a native computing platform so that kernel calls from non-native applications are translated into calls to a native kernel.

35. A method as recited in claim 34, wherein the emulating step comprises:

translating non-native CPU instructions into native CPU instructions;

translating addresses from non-native length into native length;

limiting addressable memory to a range addressable by non-native program modules.

36. A method as recited in claim 35, wherein the emulating step further comprises translating a non-native paradigm for passing parameters into a native paradigm for passing parameters.

37. A method as recited in claim 34, wherein the converting step further comprises translating words from non-native word size into native word size.

38. A computer comprising one or more computer-readable media having computer-executable instructions that, when executed by the computer, perform the method as recited in claim 34.

39. A computer-readable medium having computer-executable instructions that, when executed by a computer, performs the method as recited in claim 34.

40. A kernel emulator configured to emulate a non-native kernel for a native computing platform so that kernel calls from non-native applications are translated into calls to a native kernel.

41. An emulator as recited in claim 40, wherein the emulator comprises:

an instruction-translator configured to translate non-native CPU instructions into native CPU instructions;

an address-translator configured to translate addresses from non-native length into native length;

an memory constrainer configured to limit addressable memory to a range addressable by non-native program modules.

42. An operating system on a computer-readable medium, comprising:

a native kernel configured to receive calls from native program modules;

a kernel emulator as recited in claim 40configured to receive calls from non-native program modules.

43. A kernel emulator for non-native program modules, the emulator comprising:

target-platform determiner configured to determine a target platform of a non-native program module, wherein the target-platform determiner comprises:

an instruction-type detector configured to determine the type of non-native instructions that the non-native program module employs;

a translator selector configured to select a translator capable of translating the non-native instructions determined by the instruction-type detector into native instructions; and

at least one translator, which may be selected by the selector, configured to translate non-native instructions of the non-native program module into native instructions;

a target-platform simulator configured to simulate the selected target platform so that calls kernel calls from non-native program modules are converted into native kernel calls.

44. An operating system on a computer-readable medium, comprising:

a native kernel configured to receive calls from native program modules;

a kernel emulator as recited in claim 43 configured to receive calls from non-native program modules.

45. A kernel emulator for non-native program modules, the emulator comprising:

an interceptor configured to intercept kernel calls from non-native program modules;

a call-converter configured to convert non-native kernel calls intercepted by the interceptor into native kernel calls, wherein the call-converter comprises:

an instruction-translator configured to translate non-native CPU instructions into native CPU instructions;

an address-translator configured to translate addresses from non-native length into native length.

46. An operating system on a computer-readable medium, comprising:

a native kernel configured to receive calls from native program modules;

a kernel emulator as recited in claim 45 configured to receive calls from non-native program modules.